Topology-Aware Neighborhoods for Point-Based Simulation and Reconstruction

Florian Canezin, Gael Guennebaud, Loic Barthe

Eurographics/ ACM SIGGRAPH Symposium on Computer Animation (2016), pp. 1–11
Ladislav Kavan and Chris Wojtan (Editors)

Topology-Aware Neighborhoods for Point-Based Simulation and Reconstruction

Florian Canezin[1], Gaël Guennebaud[2], Loïc Barthe[1]

[1]IRIT - Université de Toulouse [2]Inria - Université de Bordeaux

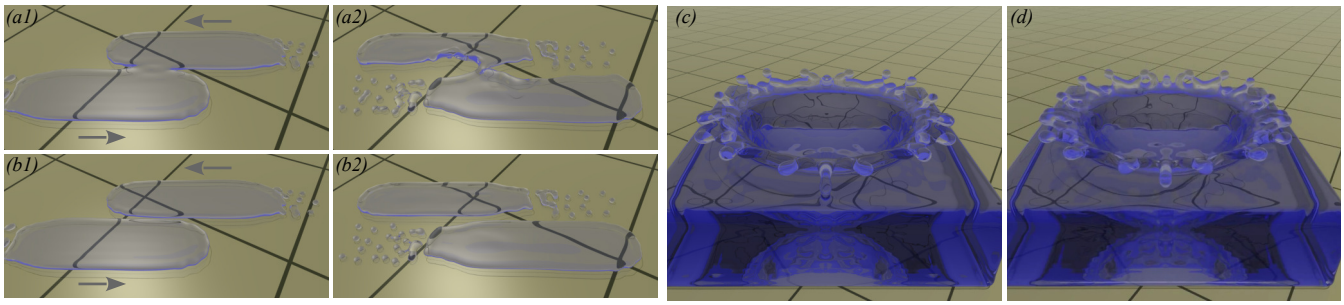


Figure 1: *Two SPH fluid simulations using a standard Euclidean particle neighborhood (a,c), and our new topological neighborhood (b,d). On the left, two fluid components are crossing while moving in opposite directions. Our new neighborhood performs accurate merging computations and avoids both unwanted fusion in the reconstruction and incorrect fluid interaction in the simulation. On the right, our accurate neighborhoods lead to different shape of the splash, and enable the reconstruction of the fluid with an adequate topology while avoiding bulging at distance.*

Abstract

Particle based simulations are widely used in computer graphics. In this field, several recent results have improved the simulation itself or improved the tension of the final fluid surface. In current particle based implementations, the particle neighborhood is computed by considering the Euclidean distance between fluid particles only. Thus particles from different fluid components interact, which generates both local incorrect behavior in the simulation and blending artifacts in the reconstructed fluid surface. Our method introduces a better neighborhood computation for both the physical simulation and surface reconstruction steps. We track and store the local fluid topology around each particle using a graph structure. In this graph, only particles within the same local fluid component are neighbors and other disconnected fluid particles are inserted only if they come into contact. The graph connectivity also takes into account the asymmetric behavior of particles when they merge and split, and the fluid surface is reconstructed accordingly, thus avoiding their blending at distance before a merge. In the simulation, this neighborhood information is exploited for better controlling the fluid density and the force interactions at the vicinity of its boundaries. For instance, it prevents the introduction of collision events when two distinct fluid components are crossing without contact, and it avoids fluid interactions through thin waterproof walls. This leads to an overall more consistent fluid simulation and reconstruction.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation.

1. Introduction

Fluid simulation for computer graphics is a very active research area for which more and more realistic solutions are expected. Among the different methods, point-based simulations, including the popular Smoothed Particle Hydrodynamics (SPH) method, were first introduced for astronomical simulations [Luc77, GM77]. They became very popular in computer graphics for their efficiency when they are applied to fluid simulations [MP89, MCG03, MSKG05, BTT09]. They are now widely used by the animation industry. In point-based simulations, the fluid is discretized with a set of particles carrying its physical properties (mass, velocity, density, etc). These properties then evolve during the simulation

according to the influence of the local environment (collision with obstacles, external forces, etc) and the particle interactions within the fluid. We refer to the survey of Ihmsen et al. [IOS*14] for a recent overview of SPH fluids.

When computing the internal interactions within the fluid, a common and critical step is the computation of a particle neighborhood that defines which particles influence the evolution of the fluid's physical properties. Although only the particles within a locally common fluid component physically interact, current neighborhood computations do not consider the fluid topology and they simply return the set of particles located at a distance lower than a certain threshold. In general, the computation of fluid interactions requires this distance to include at least three rings of neighbors around a particle. On the one hand, this Euclidean neighborhood is computationally very efficient and representative for the fluid inner parts. On the other hand, at the vicinity of the fluid boundaries, particles belonging to disconnected fluid components can be neighbors and for all of them, the physics is solved as if they are all within the same contiguous part of the fluid. This often happens when simulating flows and splashes, and it introduces incorrect behaviors in the simulation such as inappropriate rotations and fluid interactions through thin walls.

At each time step, once the simulation is computed, the fluid surface has to be reconstructed in order to be visualized. In point-based simulation, the fluid surface is in general implicitly defined as the level-set of a smooth field function [Bli82] or a distance field [ZB05, SSP07, APKG07, BGB15], which are reconstructed through 3D radially-symmetric fall-off functions attached to the particles. In these reconstructions, a neighborhood computation is also required to blend the contributions of the nearby particles, and thus produce a smooth reconstruction. When using Euclidean neighborhoods, all particles are automatically blended together to form the reconstructed surface. This method also merges close fluid components which are not in contact. This produces an inaccurate final surface, which is not exactly representative of the simulated fluid. This behavior can be minimized, but not prevented, using anisotropic fall-off functions [YT10]. To our knowledge, the only work tackling this issue proposes to group the particles globally by connected components and to blend only the particles within the same component for rendering [YT13]. In this method particles are considered in the same component if they are close enough. Even though it provides a partial solution for improving the fluid rendering, this global approach does not allow the detection of disjoint, but close parts of the same component. Neither is the method able to detect the local fusion and separation of particles in an accurate fashion nor produce the asymmetric behavior of fluid fusion and separation. Two fluid components are to be fused only when they collide, while surface tension forces maintain two fluid components connected during separation up to a split when the fluid junction becomes too thin.

Most of the research on the control of shape composition and blending from 3D field functions is found in the field of composite geometric modeling by implicit surfaces. As explained in Section 2, the management of an accurate fluid topology, adequate for both simulation and surface reconstruction is still an open, yet challenging problem.

Contribution: In this paper, inspired by the blending graphs introduced to control the compositions in soft object modeling [OM93, DG95], we propose to manage the fluid topology using a graph. The fluid topology is represented at the level of each particle by its list of neighbors within its local fluid component. The main contributions are then the temporally coherent neighborhood updates during the simulation with a detection and treatment of the particles fusion and separation together with a dedicated surface reconstruction. We take into account the asymmetric behavior of particles fusion and separation, while maintaining the coherence between the physical simulation and the reconstructed surface. Thereby, we avoid the introduction of the inconsistent particle behaviors caused by the use of Euclidean neighborhoods, and we bring a solution to a problem made extremely complex in the context of point-based simulation in which tens of thousands of particles are involved. The neighborhood management is computationally intensive and we show how computations can be efficiently accelerated and reduced (Section 6).

2. Related Work

In point-based simulations, the fluid surface is most of the time defined as an iso-surface in a 3D scalar field computed by summing all the 3D field functions representing each individual particle. This sum is the blending operator for implicit surfaces introduced by Blinn [Bli82] that produces a smooth surface from the sum of a set of positive, compactly supported, radially decreasing field functions [WMW86, MCG03]. The main limitation of this operator when applied for reconstructing a fluid surface is the blending at distance artifact. It deforms disjoint fluid components when they come in proximity and even merges them if they come closer while they are not colliding. This shape representation problem has been widely studied over the past years but no effective solution has been proposed for the very challenging case of the blending of tens of thousands of dynamic particles with the additional constraint that the fusion behavior of particles (when two fluid components collide) is different from their separation behavior.

The blending between particles is generated by the summation of their respective field contributions. Thus, the blending size is controlled by the radial slope and radius of the field functions defining the particle contributions to the fluid representation. For instance, the higher the field values and the larger the radius, the larger the blending size. A first family of approaches tries to adapt these slope and influence by particle [BS95, BGC98, WW00, HL03] so that the blending size between a particle and its neighborhood can be locally adjusted. This results in an isotropic blending behavior where a particle cannot both blend with its neighbors and not blend with others that are at proximity but not in the same fluid component. Extending this idea, an anisotropic particle field contraction or dilatation [dWv09] allows the same particle to blend differently with different particles around it. This approach is effective in the case of a low number of neighbors, when the local field deformation only influences the desired particles. This is not the case in point-based simulations where a single particle has often a very large number of neighbors. In that case, the local field modifications do not only affect the desired neighbors but also a set of nearby particles for which a different blending behavior is expected, making this approach ineffective.

In order to overcome these limitations, blending graphs have been introduced [OM93, GW95]. In these structures, particles are sorted by component such that all particles within the same component blend while those in different components collide and generate a contact surface. In this graph, components can be connected by duplicating, in each component, the particles by which they are linked. The surface is however C^0 continuous at these junctions. Desbrun and Gascuel [DG95] propose a dynamic version of this graph in which each particle stores its list of neighbors in its component. A seed-based technique for contact detection and volume preservation is then presented in order to perform contact deformations and fusion when particles collide. This seed-based update mechanism remains however computationally too expensive to be considered in point-based simulations involving tens of thousands of particles.

Recently, more advanced blending operators automatically controlling contact and blending effects between particles have been developed. Bernhardt et al. [BBCW10] detect and process the particle collisions from their mesh representation. In order to avoid the expensive mesh-based collision detection, Gourmel et al. [GBC*13] introduce gradient-controlled blending operators. While very effective in the presence of a small number of sparse particles, these operators are binary (they compose field functions by pairs) and thus they cannot control the blending of a large number of dense particles. Finally, Zanni et al. [ZCG14] adjust the blending behavior of convolution implicit surfaces [BS91] by locally scaling the field functions according to their gradient norm. This requires an homogeneous radial variation of field functions which does not hold for SPH simulations in which the radial slope of field functions varies according to the local particle density.

Currently, no blending operator is able to reconstruct a smooth fluid surface respecting the fluid topology and the asymmetry of the fusion/separation effects in a point-based simulation. Our solution to surface reconstruction solves both problems and in addition, it provides a topologically consistent update of particle neighborhoods. This is required to avoid the incorrect behavior of point-based simulations when distinct fluid components are in proximity.

3. Overview

Our neighborhood computation can be used in most point-based fluid simulation and without loss of generality, it is exposed and illustrated in an SPH simulation. Each particle of index i is associated with a position $\mathbf{p}_i$, a density ρ_i varying over time, and a constant mass m_i. For simplifying the exposition, we assume that all particles have the same radius of influence $2h$ and their radius is $h/2$. At each time step, the density and position of each particle i are updated from the set N_i of particles j within a given distance $2h$ from i, that is $N_i = \{j \mid \|\mathbf{p}_j - \mathbf{p}_i\| < 2h\}$. As motivated in the introduction, this set of Euclidean neighbor particles might improperly include particles belonging to a different component of the simulated fluid, which is problematic for both simulation and surface reconstruction. Our main objective in this work is therefore to filter these neighborhoods such that only the particles that are locally part of the same fluid component interact. As illustrated in Figure 2, we intuitively define this notion of local components by considering the pairs of particles that are directly connected by a "piece" of fluid. These so called *topological* neighborhoods $G_i \subseteq N_i$ are computed and stored for each particle i and maintained throughout the simulation such that both the simulation and visualization remain consistent.

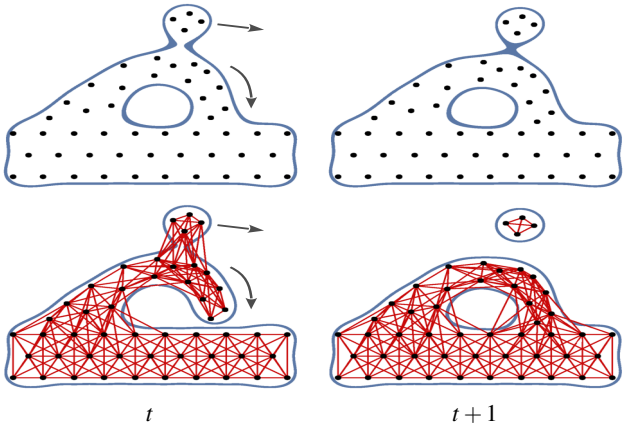


Figure 2: *Overview of our approach at two successive time steps. Particles are shown in black, and the reconstructed fluid surface is in blue. Top row: standard Euclidean neighborhood and blending by sum. Bottom row: simulation and reconstruction using our topological neighborhoods (in red), which are updated at each time step according to fusion and separation events.*

The two main steps of our algorithm are depicted in Figure 2. The top row illustrates the surface reconstruction with blending at distance obtained in the case of a standard simulation, and the bottom row shows both the surface and the graph with topology control produced with our approach. Given the set of n particles indexed by $i \in [1,n]$ and the set $\{G_i\}$ of their topological neighborhoods at time t (respectively black dots and red lines in Figure 2 bottom-left), we start by reconstructing a fluid surface having an adequate topology (in blue in Figure 2 bottom-left and Section 4). Then, the particles at the next time step $t+1$ are updated by restricting the simulation to the current topological neighborhoods $\{G_i\}$. This integration of our neighborhoods in the underlying SPH simulation is discussed in Section 6.1. From these new positions (black dots in Figure 2 bottom-right), the neighborhoods are updated by detecting merges and splits according to the surface reconstruction as detailed in Section 5 (red lines in Figure 2 bottom-right). This provides the new particles with their neighborhoods that are used for the next surface reconstruction at time $t+1$ (in blue in Figure 2 bottom-right).

4. Surface Reconstruction

In this section we assume that each particle i knows its topological neighborhood G_i. We now need to propose a surface reconstruction respecting this topological neighborhoods. Our proposition is based on the summation of field functions $f_i : \mathbb{R}^3 \to \mathbb{R}$ associated with each particle i. These field functions are standardly defined as follows [MCG03]:

$$f_i(\mathbf{x}) = \frac{m_i}{\rho_i} W\left(\|\mathbf{x} - \mathbf{p}_i\|\right), \tag{1}$$

where p_i is the particle position, m_i its mass, ρ_i its density and W is a compactly supported kernel of radius $2h$ for which we take the

following polynomial:

$$W(d) = \max\left(0, \left(1 - (\tfrac{d}{2h})^2\right)^5\right) .$$

The density ρ_i is computed by integrating the mass m_i over the topological neighborhood G_i:

$$\rho_i = \sum_{j \in \{G_i \cup i\}} m_j W(\|\mathbf{p}_i - \mathbf{p}_j\|) . \tag{2}$$

Whereas the masses, the kernel and its radius match the physical simulation, we emphasize that these densities are computed for reconstruction purpose only. They usually do not coincide with the densities computed in the physical simulation, as discussed in Section 6.1. For instance, this explains why the kernel W does not have to be normalized as any normalization factor would cancel in Equation 1.

At this stage, our goal is to avoid the reconstruction artifacts produced by the sum of all f_i, i.e., the blending at distance that generates surface attraction and unwanted connections as illustrated in Figure 3(a). We define a field function $\phi : \mathbb{R}^3 \to \mathbb{R}$ reproducing in each point $\mathbf{x} \in \mathbb{R}^3$ the sum of the particle's functions f_i that are in the same local fluid component only. To do so, for each particle i we define a field function g_i representing its blend with the particles of its topological neighborhood G_i:

$$g_i(\mathbf{x}) = f_i(\mathbf{x}) + \sum_{j \in G_i} f_j(\mathbf{x}) . \tag{3}$$

Figure 3(b) illustrates the set of such blended particles from the ones depicted in Figure 3(a). Particles i are in red and the neighbor particles in G_i are linked with a red edge. By construction, each field function g_i respects the neighborhood G_i and taking the union of these functions, for instance using $\max_i g_i$ [Sab68, Ric73], yields a reconstruction with adequate topology as shown in Figure 3(c). This Figure also illustrates that this topologically coherent reconstruction only produces C^0 continuous surfaces. This is due to the union of the functions g_i that generates sharp edges where they intersect, which is undesired for the reconstruction of a fluid surface.

By construction of our neighborhoods (see Section 5), when a particle j is in G_i then i is in G_j meaning that the volumes described by the functions g_i and g_j largely overlap each other. We thus need to slightly blend functions g_i, just enough to avoid sharp edges, without generating bulge and blending at a distance. This is done by introducing a weighted Ricci's [Ric73] blending operator defined as follow:

$$\phi(\mathbf{x}) = \left(\sum_i \frac{g_i(\mathbf{x})^s}{|G_i| + 1}\right)^{\frac{1}{s}} , \tag{4}$$

where s is the sharpness parameter controlling the amount of blending. The larger blending is obtained with ($s = 1$) and a sharp edge is generated when ($s = \infty$). The normalization factor $|G_i| + 1$ compensates the multiple occurrences of the same particle field function, say f_i, in the different g_j, thus avoiding the introduction of bulges. The multiple occurrences of field functions f_i come from the overlapping of the different neighborhoods G_j (Figure 3(b)). We found that taking $s = 20$ provides a good tradeoff for maintaining the expected topology while smoothing the edges as illustrated in Figure 3(d). The final fluid surface is then reconstructed as an iso-surface defined by $\{\mathbf{x} \in \mathbb{R}^3 \mid \phi(\mathbf{x}) = \mathcal{C}\}$, where $\mathcal{C}$ is taken such that the radius of an isolated particle is $h/2$, that is, $\mathcal{C} = W(h/2)$.

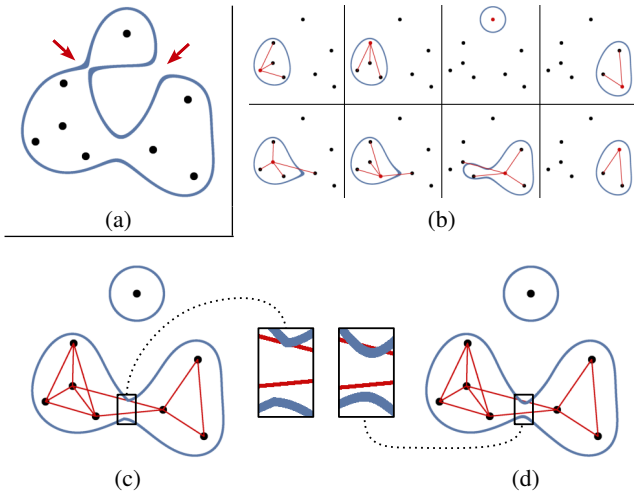


Figure 3: ***Illustration of our surface reconstruction in 2D.*** *Particles are shown with black dots, and the reconstructed surface is in blue. (a) Blending the particles with a sum leads to either unwanted blend at a distance (left red arrow) or unwanted bulging (right red arrow). (b) In our approach, each particle stores its list of topological neighbors (red connections) from which local reconstructions are defined, one per particle. (c) Taking their union yields the expected topology with sharp edges that (d) are smoothed using our weighted Ricci's operator.*

Providing a smooth fluid surface reconstruction is important for several reasons: it generates a smooth normal field well suited for shading and it enables an accurate mesh extraction through binary search along grid edges or re-projection. in addition, it enables the high-quality ray-tracing of the surface.

5. Topological Neighborhoods

Given the surface reconstruction described in the previous section, we explain how the set of topological neighborhoods $\{G_i^t\}$ at a time step t is updated from the previous simulation step $t-1$. In order to maintain an adequate topology, the main challenge is the detection of fusion and separation events in the fluid surface. Throughout these updates, we require that both the neighbor relation is symmetric: $j \in G_i \Leftrightarrow i \in G_j$ and the topological neighborhoods are consistent with respect to the fluid components, i.e., particles that are within the kernel support of each other and that are part of the same local fluid component must remain topological neighbors even though a split or non-merge event has been detected. This property boils down to the following local transitive-closure of the neighbor relation:

$$\begin{aligned} &\forall i, j \text{ s.t. } i \neq j \text{ and } \|\mathbf{p}_i - \mathbf{p}_j\| < 2h, \\ &\text{if } \exists k \in G_i^t \cap G_j^t \text{ s.t. } \max(\|\mathbf{p}_i - \mathbf{p}_k\|, \|\mathbf{p}_j - \mathbf{p}_k\|) \leq \alpha h \\ &\text{then } i \in G_j^t \text{ and } j \in G_i^t . \end{aligned} \tag{5}$$

In contrast to classical transitive-closure, our local variant restricts the transitivity condition in two ways. It only applies to the pairs of

particles that are less than $2h$ apart (first line in Equation 5), and it can be inferred only from existing pairs of particles that are close enough to each other. This proximity is controlled by the parameter α in the second line of Equation 5. We control this proximity because in SPH simulation and reconstruction, the kernel support of the field functions is usually very large, it approximately matches a three-ring neighborhood. Without this restriction, a transitive-closure would connect distant particles that would not be connected by the surface reconstruction, which is inconsistent. A typical example is the one presented in Figure 2 for which connections would be created across the fluid handle. Our experiments shown that taking $\alpha = 5/4$ is an effective choice.

Since any neighbor change modifies the reconstructed surface, satisfying all these constraints might lead to a chicken-egg problem. Our solution to avoid this involves the following three steps which are summarized in Algorithm 1.

Algorithm 1 SIMULATION STEP

1: $\{\mathbf{p}_i^t\} \leftarrow$ sph_update($\{\mathbf{p}_i^{t-1}\}, \{\rho_i^{t-1}\}, \{G_i^{t-1}\}$)
2: $\{\rho_i^t\} \leftarrow$ reconstruction_density_update($\{\mathbf{p}_i^t\}, \{G_i^{t-1}\}$)

Merging stage:
3: $\{G_i'\} \leftarrow$ merge_update($\{\mathbf{p}_i^t\}, \{\rho_i^t\}, \{G_i^{t-1}\}$)
4: local_transitive_closure($\{G_i'\}$)
5: $\{\rho_i^t\} \leftarrow$ reconstruction_density_update($\{\mathbf{p}_i^t\}, \{G_i'\}$)

Splitting stage:
6: $\{G_i^t\} \leftarrow$ split_update($\{\mathbf{p}_i^t\}, \{\rho_i^t\}, \{G_i'\}$)
7: local_transitive_closure($\{G_i^t\}$)
8: $\{\rho_i^t\} \leftarrow$ reconstruction_density_update($\{\mathbf{p}_i^t\}, \{G_i^t\}$)
9: surface_reconstruction($\{\mathbf{p}_i^t\}, \{\rho_i^t\}, \{G_i^t\}$)

Firstly: the particle positions are updated through the SPH simulation routine using the neighborhoods $\{G_i^{t-1}\}$ (Algorithm 1 line 1). For the initial time step, these neighborhoods are initialized with all particles within the radius of influence $2h$ (i.e., $G_i^0 = N_i^0$). As the particles have moved, their surface reconstruction densities have to be updated using Equation 2 (Algorithm 1 line 2).

Secondly: particle fusions are detected among the pairs of particles whose supports intersect and which are not already neighbors in $\{G_i^{t-1}\}$ (Algorithm 1 line 3). As detailed in Section 5.1, this step yields intermediate neighborhoods $\{G_i'\}$. Before going any further, these neighborhoods have to be completed to satisfy the local transitive-closure property (Algorithm 1 line 4), and surface reconstruction densities have to be recomputed to take into account the novel connections (Algorithm 1 line 5).

Thirdly: pairs of particles which are connected in $\{G_i'\}$ but that appear to be locally disconnected with respect to the fluid surface are removed (Algorithm 1 line 6) as detailed in Section 5.2. Again, the local transitive-closure property has to be ensured (Algorithm 1 line 7) to obtain the final updated neighborhoods $\{G_i^t\}$. Those are used to update the densities one more time (Algorithm 1 line 8) before to perform the surface reconstruction (Algorithm 1 line 9).

In the following, we detail how merges and splits are efficiently detected and explain in Section 5.4 how our approach is extended to ensure temporal coherence during fusions.

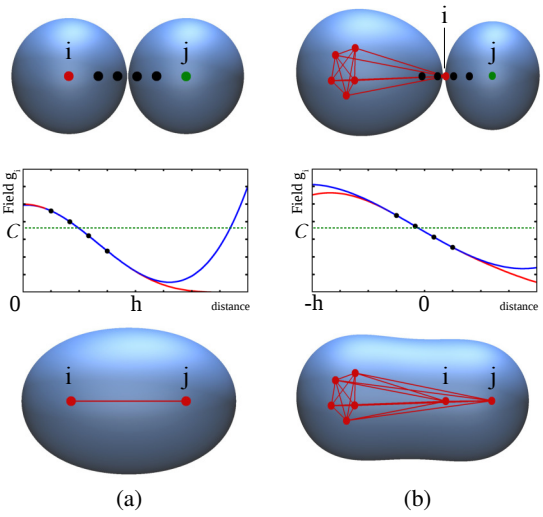


Figure 4: *Illustration of the merge detection mechanism between two primitives i and j formed by the red and green particles respectively. Top row: the current particles and reconstruction. Middle row: the plot of the field function g_i of the particle i (red curve), and its local approximation by a cubic polynomial (blue curve) passing through the four sampled positions (black dots). Bottom row: novel connections and reconstruction after merging. Two cases are depicted: (a) the position of the iso-contour is found using the first tested range, (b) whereas for a uneven sampling, a second range has to be tested.*

5.1. Component Fusion

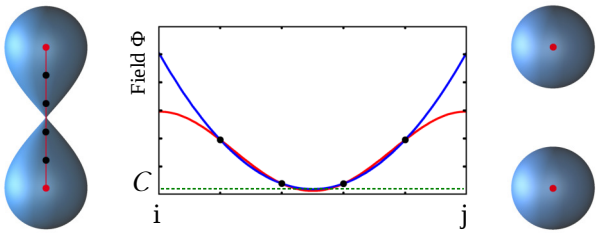
For each particle i, we compute an intermediate neighborhood G_i' as the union of the topological neighborhood G_i^t and all other particles j within its kernel support for which their respective local fluid components collide or interpenetrate. We thus consider each pair of particles i-j such that $j \notin G_i^{t-1}$, $i \notin G_j^{t-1}$ and $\|\mathbf{p}_i - \mathbf{p}_j\| \leq 2h$. Since our reconstructed surface is very close to the union of the blended neighbor particles (i.e., $\phi \approx \max_i g_i$), we can assume that each of the two blended neighborhoods g_i and g_j well represent the local fluid component around particles i and j respectively. Our problem is then to detect whether these two pieces of fluid intersect.

We detect fusion by searching along a 1D parametric line connecting the two particles. Let r_{ij} be the signed distance between $\mathbf{p}_i$ and the fluid surface defined in g_i in the direction of $\mathbf{p}_j$, that is, r_{ij} is the largest real value such that $g_i\left(\mathbf{p}_i + r_{ij}\frac{\mathbf{p}_j - \mathbf{p}_i}{\|\mathbf{p}_j - \mathbf{p}_i\|}\right) = \mathcal{C}$. By defining r_{ji} analogously, our fusion condition becomes:

$$\|\mathbf{p}_i - \mathbf{p}_j\| < \beta(r_{ij} + r_{ji}),$$

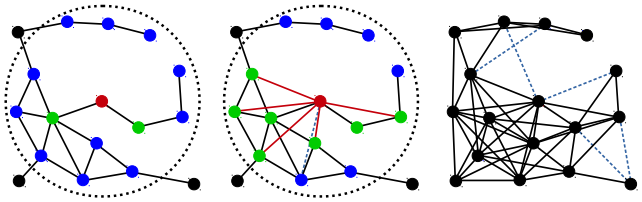
where β is a small tolerance factor compensating for the small blending produced by Equation 4 and favoring early over late fusions. We always use $\beta = 1.01$.

Since each field function evaluation has a computational cost, we estimate the values r_{ij} with cubic polynomial approximations. Even though quadratic polynomial interpolation would ease the subsequent root finding, the use of cubic interpolation is required since, as depicted in Figure 4, the scalar field along the 1D ray is expected



(a) (b) (c)

Figure 5: *Detection of a split event between two connected particles i and j. (a) Current configuration and reconstruction. (b) Plot of the max of the two field functions g_i, g_j along the segment i-j (red), and its approximation by a quadratic polynomial fitted on four sample points (black dots). (c) Since the minimum is below $\mathcal{C}$, the pair is split, resulting in disconnected particles.*



(a) (b) (c)

Figure 6: *Local transitive-closure. (a) A given particle in red with its current neighbors in green, its kernel support (large circle), and the set of particles that can be potentially connected to it in blue. (b) The first sweep of transitive-closure update for this given particle yields the addition of the five connections in red. The dashed blue line indicates a potential connection that has not been already established because their respective shared particle was too far away from them. This one will be established during the next sweep. (c) Resulting connections after applying this step to all particles repeatedly until convergence is achieved. Again, the dashed blue lines indicate a few connections that are not introduced (on purpose), because the edges that could infer them are above our threshold length αh.*

to exhibit an inflection point, which cannot be reproduced with a quadratic polynomial. As depicted in Figure 4 left, we first consider the interval $[h/4, 3h/4]$ along the given parametric line, and construct the cubic polynomial that interpolates four sample values of g_i uniformly taken within this range. In most cases, this strategy succeeds in providing the expected result. In some rare cases, as in Figure 4 right, an uneven sampling of the particle might significantly shrink the surface. The current particle might even lie outside its own local fluid component. In this case, the first sampled value is below the isovalue (i.e., $g_i(h/4) < \mathcal{C}$), and the search interval is shifted to $[-h/4, h/4]$. The same fitting procedure is then applied. Finally, if the first sampled range is within the fluid component (i.e., $g_i(3h/4) > \mathcal{C}$) then the particle cannot be at the boundary of the component, and no merge is explicitly detected for this pair. A connection might eventually be established later through transitive-closure as explained in Section 5.3.

We emphasize that this detection of fusion does not depend on the look-up order of the particles as all primitive evaluations are based on the fixed neighborhoods $\{G_i^{t-1}\}$, whereas newly detected neighbors produce $\{G_i'\}$. Implementation-wise, the use of four sample values to fit the cubic polynomial permits to fully exploit the SIMD vector instruction sets of current CPUs: these four evaluations are carried out at the cost of a single evaluation.

5.2. Component splitting

Component separation or split occurs when particles of the same fluid component move apart from each other. Each pair i-j of neighbor particles (i.e., $j \in G_i'$ and $i \in G_j'$) is checked in case splitting is required once all fusions have been performed. Two neighbor particles i and j are split only if the segment $[\mathbf{p}_i, \mathbf{p}_j]$ joining them has a part lying outside the local fluid component defined by the union of their respective blended neighborhoods g_i and g_j (see Figure 5).

More formally, let $\bar{g}_{ij}$ be the minimum of $\max(g_i, g_j)$ along the segment $[\mathbf{p}_i, \mathbf{p}_j]$. If $\bar{g}_{ij} < \mathcal{C}$, then the pair i-j is split. Otherwise, the particle i (resp. j) is inserted into G_j^t (resp. G_i^t). In practice, we quickly estimate $\bar{g}$ by fitting an univariate quadratic polynomial to a given number of sample values of $\max(g_i, g_j)$ uniformly taken in the segment $[\mathbf{p}_i, \mathbf{p}_j]$, as illustrated in Figure 5. As for detecting fusions, we found that taking four samples is accurate enough in practice while enabling fast SIMD evaluations.

This procedure requires every pair of connected particles to be tested, which is very expensive as the number of such pairs is two orders of magnitude larger than the number of particles. The number of splitting tests can be drastically reduced by observing that if two connected particles are close enough to each other, then a separation is very unlikely to occur. Each connected pair i-j such that $\|\mathbf{p}_i - \mathbf{p}_j\| < \alpha h$ are thus preserved and ignored by the splitting test, where $\alpha = 5/4$ as for the local transitive-closure in Equation 5 because it plays the same role.

We can now take advantage of local transitive-closure property to further reduce the number of splitting tests. Indeed, given a connected pair of particles i-j that can be potentially split, if there exists a third particle k satisfying the local transitive-closure property of Equation 5, then we know that the pairs i-k, and j-k will not be split, and thus the pair i-j has to be preserved bypassing the splitting test.

Finally, the condition for a pair i-j, $i \in G_j'$ and $j \in G_i'$, to be inserted into G_i^t and G_j^t can be summarized as follows:

$$\begin{array}{ll} & \|\mathbf{p}_i - \mathbf{p}_j\| < \alpha h \\ \text{or} & \exists k \in G_i' \cap G_j' \text{ s.t. } \max(\|\mathbf{p}_i - \mathbf{p}_k\|, \|\mathbf{p}_j - \mathbf{p}_k\|) \leq \alpha h \\ \text{or} & \bar{g}_{ij} \geq \mathcal{C} \end{array}$$

5.3. Transitive-closure

As explained at the beginning of this section, local transitive-closure (Eq. 5) of our topological neighborhoods has to be satisfied before their use for density estimation or local surface reconstruction. This explains why passes of transitive-closure update have to be performed both after detecting merges and splits (lines 4 and 7 of Algorithm 1). We also emphasize that the primary role of transitive-closure is to ensure that two close-enough particles lying nearby

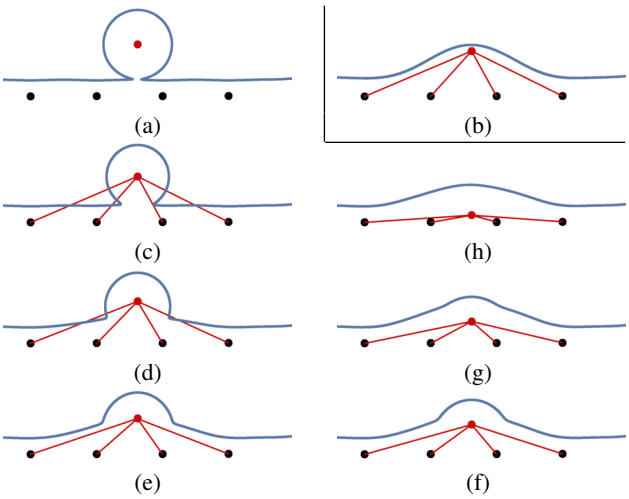


Figure 7: *Illustration of our temporal coherence mechanism on a falling particle (in red) entering in contact with another fluid component (a). (b) The newly established connections lead to a quick change in the reconstructed iso-contour. The sequence (c) to (h) shows the progressive absorption produced by our temporal weighting.*

a concave part of the fluid surface are properly connected. Otherwise, such pairs of particles would be checked for fusion several times until they become really close or that the local piece of surface becomes not too concave so that the segment lying them is entirely within the reconstructed fluid component. Therefore, our transitive-closure rule does not only improve consistency, but also the performance of the overall algorithm.

Transitive-closure is usually computed through repeated depth-first or breath-first traversals, however in our context, the number of pairs that can be added through local transitive-closure is considerably smaller than the number of existing pairs. Therefore, we found that a much faster strategy consists of looping over each pair of potentially miss-connected particles, that is, each pair i-j such that $j \notin G_i^{t-1}$ and $\|\mathbf{p}_i - \mathbf{p}_j\| \leq 2h$, and search for a common and close enough neighbor particle. This step has to be repeated until convergence is achieved, that is, until no novel connection is established. This procedure is illustrated in Figure 6. Notice that thanks to our double locality restriction, the cavity is well preserved.

During the merging stage we loop over the set of potentially missing pairs using the following method. Each pair for which no merge has been detected is appended to a list. Then, during transitive-closure updates, it is enough to loop over this list from which a pair is removed if and only if it is added to the set of neighborhoods. For the second pass of transitive-closure (after the splitting step), it is enough to consider only the list of pairs which have been split.

5.4. Temporal coherence

During a fusion near to a particle i, new particles are inserted into its topological neighborhood. These new particles are usually close to i meaning that they immediately exhibit a significant contribution to both the density ρ_i and the blended neighborhood g_i of the given particle i. As a result, popping might occur in the reconstructed surface, as show in Figures 4 and 7. In the second figure, a falling droplet gets immediately absorbed when contact is detected. We address this issue by tracking the "age" a_{ij} of each neighbor relation. For each newly connected particle i-j, a_{ij} is initialized to zero and updated at each frame as follows:

$$a_{ij}^t = \min\left(1, a_{ij}^{t-1} + \Delta t/\gamma\right) \tag{6}$$

where Δt is the time in seconds between two frames, and γ is the duration in which the age of the relation saturates to 1. For the first frame, the age is initialized to one for all pairs (i.e., $a_{ij}^0 = 1$). This age is used in the computation of both the densities and the local reconstructions as follows:

$$\rho_i = \sum_{j \in \{G_i \cup i\}} a_{ij} m_j W(\|\mathbf{p}_i - \mathbf{p}_j\|), \tag{7}$$

$$g_i(\mathbf{x}) = f_i(\mathbf{x}) + \sum_{j \in G_i} \left(1\text{-}(1\text{-}a_{ij})^3\right) f_j(\mathbf{x}). \tag{8}$$

The difference in weighting enables a better balance between the temporal variations of the density versus blending. The behavior produced by such a temporal weighting scheme is depicted in Figure 7.

6. Practical implementation

6.1. Integration in a point-based simulation

Integrating our approach within an existing simulation code only requires to replace loops over Euclidean neighbors by loops over our topological ones. We implemented our prototype using DualSPHysics [CDR*15] for the particle simulation, for which we enabled the Shepard density filter for adjusting densities at the vicinity of the fluid surface. From the physical aspect, some precautions must however be taken.

The standard SPH integration kernel assumes that the ambient space is full of particles whereas in general, only fluid particles are simulated. This results in a bias in the density estimation at the proximity of the fluid surface and removing particles from an Euclidean neighborhood in these areas, even though they belong to a separated fluid component, leads to physical incorrectness in the density computations. This situation changes as soon as such under-resolved particle neighborhoods is numerically compensated, using for instance adjusted integration kernels [BK02]. In that case, the use of our topological neighborhood allows computing densities and forces consistently and independently of the presence of a nearby disconnected fluid component, and inadequate fluid interference from disconnected components are avoided. We also point out that the use of our topological neighborhood would naturally handle very thin walls between fluids as particles in each side would belong to its own topological fluid component. This avoids particles interactions through the wall and only wall-particle interactions remains to be simulated. We do not show such an example case because the DualSPHysics fluid simulation enforces the use of large enough walls to avoid the unexpected fluids interactions, thus preventing particles on one side of the wall to come close enough to penetrate the influence radius of particles on the other side.

6.2. Topological neighborhood implementation

Neighborhood updates (steps 2 to 8 of Algorithm 1) are implemented on the CPU. All these steps but the transitive-closure passes are accelerated taking advantage of multi-threading with OpenMP. In order to avoid memory reallocation during neighborhood updates, each particle stores the list of all neighbors within the range $2h$. Each list is kept sorted with respect to indices and such that topological neighbors appear first. These lists are updated using a 3D grid after the particle positions have been updated. Flags are used to distinguish between the different stages of the update (G^{t-1}, G', G^t). Sorted lists enable fast searches and set intersections during transitive closure updates.

In order to maintain numerical accuracy, the physics solver, such as DualSPHysics, usually subdivides the target time-step into numerous sub-timesteps according to the different parameters of the simulation. This explains why in Table 1, the larger "Splash" simulation appears to be faster than the smaller "Table" simulation. Following this strategy, we propose to update the neighbor lists for every main timestep, and in the sub-timesteps whenever a particle ranges a too long distance since the last neighborhood update. This maintains a consistent topology during the simulation while reducing the cost of a systematic detection of fusion and split events.

Scene		**Table**		**Splash**
#particles		3.8k	22.8k	56k
SPH simulation		0.11	1.47	0.3
$\{G_i\}$ **update**		0.02	0.17	0.49
GPU Eval	*Grid resolution*	462×264×216		216^2×334
	no optimization	*0.68*	*0.83*	*66.33*
	+ early stop	*0.68*	*0.84*	*14.09*
	+locality of adjusted neighborhoods	*0.40*	*0.70*	*1.29*
	Final eval time *(+CUDA warps)*	0.33	0.54	0.99
	Standard sum of the f_i	*0.15*	*0.23*	*0.28*

Table 1: *Average timings in seconds for the update of one frame for the two scenes shown in Figure 1. Reported timings include the SPH simulation using the DualSPHysics library, the update of our topological neighborhoods $\{G_i\}$ on the CPU, and the filling of the full marching cube grid on the GPU using either our reconstruction method or a standard sum of the f_i. The timings for our reconstruction method are reported with different level of optimization, starting from the naive version evaluating the full field function ϕ everywhere, then successively adding early-stop, the restriction to area containing adjusted neighborhoods, and finally the CUDA warp coherence.*

6.3. Efficient surface evaluation

Our reconstructed iso-surface is extracted as a mesh from a uniform grid which is filled by evaluating ϕ through a CUDA implementation. Each evaluation of $\phi(\mathbf{x})$ (Eq. 4) involves a pair of nested loops on each nearest primitive and each particle of the current primitive. These loops are required to find all particles i such that $g_i(\mathbf{x})$ is non null, i.e., we have to find all particles within a sphere centered at $\mathbf{x}$ and of radius $4h$.

In practice, this search and evaluation can be greatly accelerated by reducing this radius. Indeed, due to the use of exponentiation with a large power, that is g_i^{20} in Equation 4, the contribution of a given field function g_i quickly becomes negligible when moving away, and only the largest ones have a real impact on the result. Therefore, thanks to the very large overlap between the field functions g_i, we found that it is always sufficient to consider only the particles within a radius of $3.25h$ for a gain of about $\times 1.5$. Since the iso-surface are expected to occur at a distance $h/2$ of the particles, this is a rather conservative choice, and the search radius can be aggressively shrunk without impacting the reconstruction.

A second optimization consists in stopping the sum over the primitives as soon as it exceeds $\mathcal{C}^s$, meaning that the evaluation point $\mathbf{x}$ is within the fluid. As shown in Table 1, this *early stop* optimization significantly reduces the grid filling cost, especially where the fluid covers a large volume.

In addition, our computation of the global field ϕ is only required in the vicinity of adjusted neighborhoods. Everywhere else it is enough to sum over the particle's field f_i of the Euclidean neighbors. Finally, the number of overall evaluations can be greatly reduced by evaluating the field function at the proximity of surface particles, as explained by Akinci et al. [AIAT12].

When implementing the grid filling on a GPU, additional care must be taken to maximize parallelism among the threads of the same warp, that is, among the packet of typically 32 threads that follow the same execution flow. Indeed, because of the nested loops, threads attached to nearby grid points can quickly diverge. We enforce a coherent evaluation by attaching blocks of $4\times4\times2$ grid points to the same warp that performs a common traversal of the grid to query the primitives within the union of individual queries. This strategy yields an additional $\times 1.5$ speedup, even though branch divergence still exist because some primitives which have to be processed by at least one thread might have to be skipped for the others. Further acceleration is thus obtained by skipping the farthest primitives in a coherent manner using the following pseudo-code algorithm where $\{i_1,\ldots,i_m\}$ denotes the set of primitive indices processed by the current wrap:

$k = 1$
while $k \leq m$ **do**
  while $k \leq m$ **and** $\|\mathbf{x}-\mathbf{p}_i\| > 3.25h$ **do** $k = k+1$ **;**
  if $k \leq m$ **then** accumulate the contribution of g_{i_k} **;**
  $k = k+1$

This algorithm has the effect of re-synchronizing the threads by making them wait until all threads have a primitive to work on. This yields an additional $\times 1.5$ speedup factor for large grid resolution.

7. Results and limitations

We have evaluated our approach by comparing our results with those obtained by using Euclidean neighborhoods for the SPH simulation, and the sum of the field functions f_i for the surface reconstruction (denoted as standard). Our test scenes are shown in Figure 1.

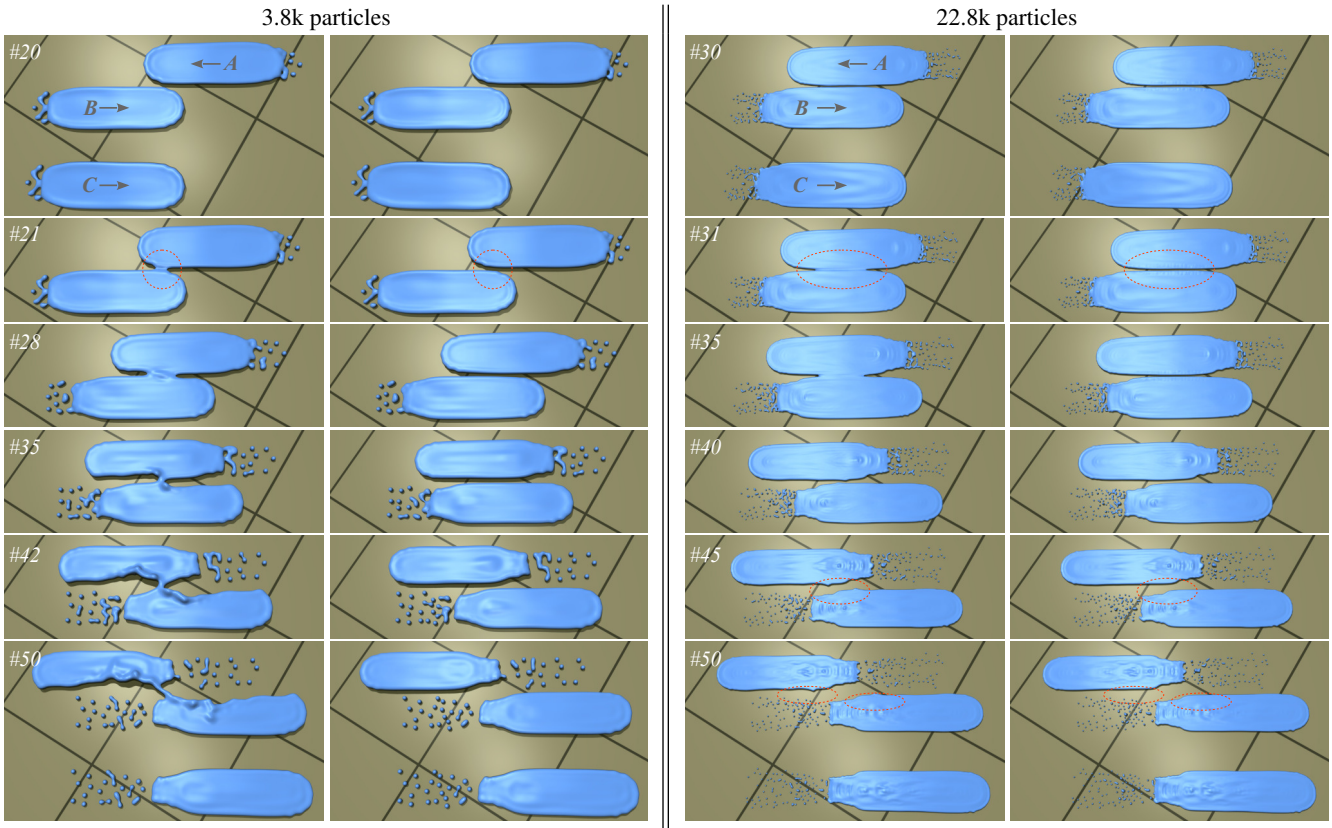


Figure 8: *Short sequences of the "Table" simulations using either Euclidean or our topological neighborhoods. The white labels indicate the respective frame number. The two fluid components A and B incorrectly interact and merge when using Euclidean neighborhoods, while they naturally cross without interacting when using our approach (component B remains identical to its isolated copy C).*

Quality

The first scene, called "Table" (Figure 1-left), consists of two pieces of water labeled A and B, moving in opposite directions along a flat table, and passing nearby each other. The particles of each component have been initialized such that the two respective fluids do not intersect if simulating them separately. For evaluation purpose, we added an isolated copy of the second component labeled C, which is initialized with the exact same conditions. Figure 8 shows short sequences for two different simulation resolutions. When using 1260 particles per component, Euclidean neighborhoods quickly merge the two nearby components in both the simulation and reconstruction, creating small splashes. As a result, the components becomes significantly distorted. In contrary, our topological neighborhoods properly resolve the reconstruction ambiguity and thus prevent the interaction between the two disconnected components within the simulation. As a result, the component B remains identical to its isolated duplicate C. The effect of distant interactions of Euclidean neighborhoods can be diminished by increasing the density of particles. Nonetheless, as shown in Figure 8-right, even after increasing the number of particles by a factor 6, some unwanted fusions and distortions are still present.

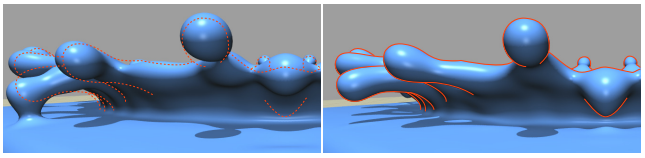


Figure 9: *Illustration of the diverging behavior between a standard simulation (left), and our approach (right). To highlight the simulation differences, the surface of both simulations have been reconstructed using Euclidean neighborhoods, and the silhouette of the right image is reported to the left one.*

In the second test scene (Figure 1-right), a droplet composed of about 180 particles hits a box of still water producing a splash. As seen in Figure 9, the two simulations exhibit differences. In order to ease the evaluation of our reconstruction method in Figure 10, we thus compare it to reconstruction results obtained using a standard sum of the f_i over the Euclidean neighbors, but using the same particle simulation as in our method. Unwanted bulging and merging at a distance can be observed throughout the sequence produced by the Euclidean neighborhoods, whereas our approach success-

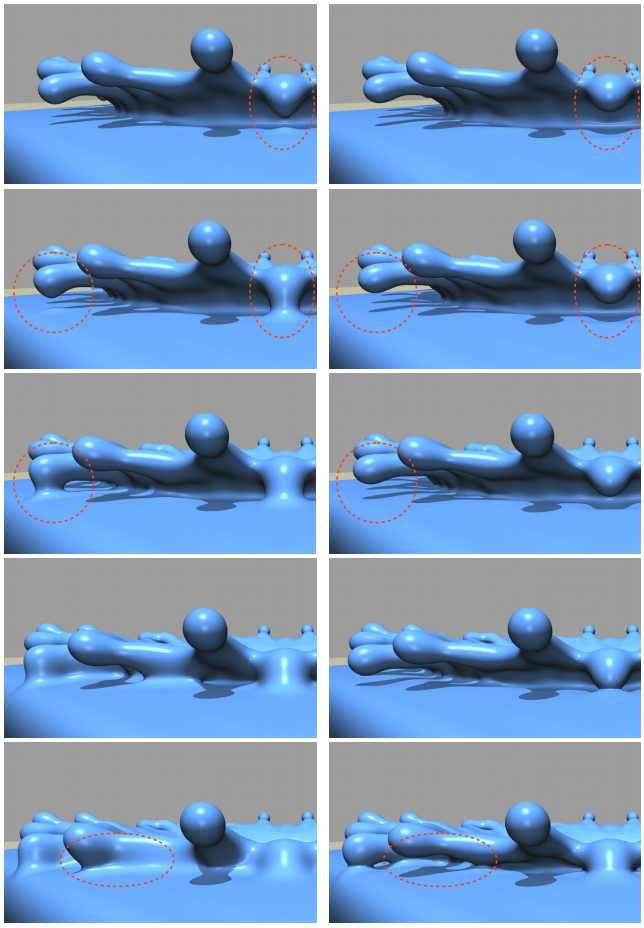


Figure 10: *Five consecutive frames of the "Splash" simulation using our neighborhoods. This figure compares the reconstruction with a standard Euclidean sum of the field functions f_i (left), to our reconstruction method (right). Notice how the water surface is deformed and merged prior to the actual contact event when using Euclidean neighborhoods.*

fully tracks the expected topology of the fluid until components get very close to each other. Figure 11 shows the effect of our temporal merging mechanism: as expected, it can be seen that the falling droplets get smoothly absorbed by the larger fluid component.

Performance

We have measured the performance of our prototype on a computer equipped with an Intel Core-i7@3.4GHz and a Geforce 580 GTX. Table 1 reports average costs of the different steps of our algorithms for three different simulations. This table also details the impact of the aforementioned optimizations when filling a grid on the GPU for our simulations. It can be seen that the relative overhead to update our topological neighborhood highly depends on the simulation. Indeed, for the "Table" scene, the overhead of our approach is marginal because this scene requires several intermediate simulation steps to avoid numerical instabilities, as automatically determined by DualSPHysics. On the other hand, DualSPHysics

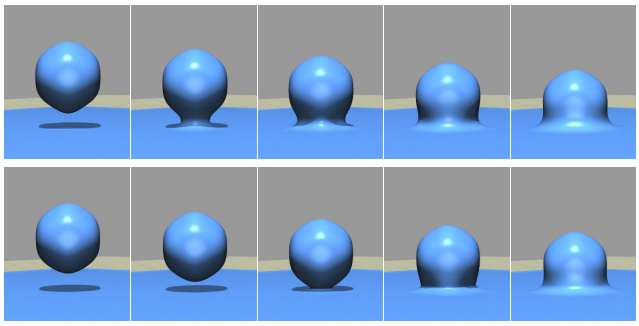


Figure 11: *Comparison of the standard approach (top row), with our approach (bottom row) on the "Splash" simulation. Notice how our approach successfully resolve the contact event while enabling a smooth transition through temporal weights. The falling droplet is composed of about* 180 *particles.*

can simulate the "Splash" scene at a much higher rate, even though it contains more particles. For this scene, our neighborhood update is of the same order as the SPH simulation itself.

Regarding the reconstruction step, thanks to our various optimizations, the overhead of our approach compared to a standard sum over the Euclidean neighbors ranges from a factor $\times 2$ to $\times 3$.

Limitations

The detection of fusion and separation relies on some heuristics, meaning that they can be detected slightly too early or too late. Nonetheless, the effects of these approximations are seldom perceptible and they are considerably less prominent than in standard approaches.

In this work we focused on the preservation of the fluid topology, although we have neglected the fairness of the final fluid surface. Although mesh based smoothing techniques can be used, it would be interesting to investigate the extension of our method to take advantage of recent advances in the reconstruction of surfaces with better tension properties from SPH simulations. For instance, one could incorporate anisotropic primitives [YT10] within our method. Another approach would be to define the individual clusters g_i using any implicit reconstruction methods, for instance Solenthaler et al.'s smooth distance field [SSP07], convert it to a local field function [CGB13], and combine them using the original Ricci's blending operator [Ric73].

8. Conclusion and future work

We introduced a novel neighborhood computation mechanism that preserves the topology of the animated fluid in point-based simulations. Our neighborhoods can be trivially integrated into an existing point-based fluid simulation system as it simply replaces the classical Euclidean neighborhood. We also show how the new neighborhood can be efficiently exploited to reconstruct a fluid surface avoiding unwanted blends and bulges, while respecting the topology of the fluid. We also take into account subtle effects such as the asymmetric fluid behavior when fluid components merge or split. Overall, we provide a neighborhood computation capturing the fluid topology together with the way it can be used coherently

in both the simulation and the surface reconstruction. Finally, our results show that our neighborhood solves some incorrect behaviors in the simulation and can lead to significantly different particle motions.

As future works, we would like to continue to diminish the computational overhead brought by the use of our neighborhoods, for instance by exploring a GPU implementation of their update, and alternative reconstruction strategies exhibiting a lower algorithmic complexity. We would also like to investigate how to integrate fluid surface tension efficiently into our method.

Acknowledgments

This work was partially funded by the IM&M project (ANR-11-JS02-007) and the CIMI Labex (ANR-11-LABX-0040-CIMI within the program ANR-11-IDEX-0002-02).

References

[AIAT12] AKINCI G., IHMSEN M., AKINCI N., TESCHNER M.: Parallel Surface Reconstruction for Particle-Based Fluids. *Computer Graphics Forum* (2012). 8

[APKG07] ADAMS B., PAULY M., KEISER R., GUIBAS L. J.: Adaptively sampled particle fluids. *ACM Trans. Graph. 26*, 3 (2007). 2

[BBCW10] BERNHARDT A., BARTHE L., CANI M.-P., WYVILL B.: Implicit blending revisited. *Proc. of Eurographics, Computer Graphics Forum 29*, 2 (2010), 367–376. 3

[BGB15] BHATTACHARYA H., GAO Y., BARGTEIL A.: A level-set method for skinning animated particle data. *IEEE Transactions on Visualization and Computer Graphics 21*, 3 (2015), 315–327. 2

[BGC98] BARTHE L., GAILDRAT V., CAUBET R.: Combining implicit surfaces with soft blending in a CSG tree. In *Proc. of CSG Conference Series* (1998), pp. 17–31. 2

[BK02] BONET J., KULASEGARAM S.: A simplified approach to enhance the performance of smooth particle hydrodynamics methods. *Applied Mathematics and Computation 126*, 2-3 (2002), 133–155. 7

[Bli82] BLINN J. F.: A generalization of algebraic surface drawing. *ACM Trans. Graph. 1*, 3 (1982), 235–256. 2

[BS91] BLOOMENTHAL J., SHOEMAKE K.: Convolution surfaces. In *ACM SIGGRAPH '91* (1991), vol. 25, pp. 251–256. 3

[BS95] BLANC C., SCHLICK C.: Extended field functions for soft objects. In *Proc. of Implicit Surfaces 1995* (1995), pp. 21–32. 2

[BTT09] BECKER M., TESSENDORF H., TESCHNER M.: Direct forcing for lagrangian rigid-fluid coupling. *IEEE Trans. Vis. Comput. Graph. 15*, 3 (2009), 493–503. 1

[CDR*15] CRESPO A., DOMÍNGUEZ J., ROGERS B., GÒMEZ-GESTEIRA M., LONGSHAW S., CANELAS R., VACONDIO R., BARREIRO A., GARCÍA-FEAL O.: Dualsphysics: Open-source parallel {CFD} solver based on smoothed particle hydrodynamics (sph). *Computer Physics Communications 187*, 0 (2015), 204 – 216. 7

[CGB13] CANEZIN F., GUENNEBAUD G., BARTHE L.: Adequate Inner Bound for Geometric Modeling with Compact Field Function. *Computer & Graphics (proceedings of SMI 2013) 37*, 6 (2013), 565–573. 10

[DG95] DESBRUN M., GASCUEL M.-P.: Animating soft substances with implicit surfaces. In *Proceedings of ACM SIGGRAPH '95* (1995), pp. 287–290. 2, 3

[dWv09] DE GROOT E., WYVILL B., VAN DE WETERING H.: Locally restricted blending of blobtrees. *Computers & Graphics 33*, 6 (2009), 690–697. 2

[GBC*13] GOURMEL O., BARTHE L., CANI M.-P., WYVILL B., BERNHARDT A., PAULIN M., GRASBERGER H.: A gradient-based implicit blend. *ACM Trans. Graph. 32*, 2 (2013), 1–12. 3

[GM77] GINGOLD R. A., MONAGHAN J. J.: Smoothed particle hydrodynamics: theory and application to non-spherical stars. *Monthly Notices of the Royal Astronomical Society 181*, 3 (1977), 375–389. 1

[GW95] GUY A., WYVILL B.: Controlled blending for implicit surfaces using a graph. In *Proc. of Implicit Surfaces 1995* (1995), pp. 107–112. 3

[HL03] HSU P. C., LEE C.: Field functions for blending range controls on soft objects. *Proc. of Eurographics, Computer Graphics Forum 22*, 3 (2003), 233–242. 2

[IOS*14] IHMSEN M., ORTHMANN J., SOLENTHALER B., KOLB A., TESCHNER M.: SPH Fluids in Computer Graphics. In *Eurographics 2014 - State of the Art Reports* (2014). 2

[Luc77] LUCY L.: A numerical approach to the testing of the fission hypothesis. *Astronomical Journal 82* (1977), 1013–1024. 1

[MCG03] MÜLLER M., CHARYPAR D., GROSS M.: Particle-based fluid simulation for interactive applications. In *Proceedings of SIGGRAPH/Eurographics Symposium on Computer Animation* (2003), pp. 154–159. 1, 2, 3

[MP89] MILLER G., PEARCE A.: Globular dynamics: A connected particle system for animating viscous fluids. *Computer & Graphics 13*, 3 (1989), 305–309. 1

[MSKG05] MÜLLER M., SOLENTHALER B., KEISER R., GROSS M.: Particle-based fluid-fluid interaction. In *Proceedings of SIGGRAPH/Eurographics Symposium on Computer Animation* (2005), pp. 237–244. 1

[OM93] OPALACH A., MADDOCK S.: Implicit surfaces: Appearance, blending and consistency. In *In Fourth Eurographics Workshop on Animation and Simulation* (1993), pp. 233–245. 2, 3

[Ric73] RICCI A.: A constructive geometry for computer graphics. *Computer Journal 16*, 2 (1973), 157–160. 4, 10

[Sab68] SABIN M.-A.: The use of potential surfaces for numerical geometry. In *Tech. Report VTO/MS/153, British Aerospace Corp., Weybridge, U.K.* (1968). 4

[SSP07] SOLENTHALER B., SCHLÄFLI J., PAJAROLA R.: A unified particle model for fluid-solid interactions. *Computer Animation and Virtual Worlds 18*, 1 (2007). 2, 10

[WMW86] WYVILL G., MCPHEETERS C., WYVILL B.: Data structure for soft objects. *The Visual Computer 2*, 4 (1986), 227–234. 2

[WW00] WYVILL B., WYVILL G.: Better blending of implicit objects at different scales. *ACM Siggraph 2000 presentation* (2000). 2

[YT10] YU J., TURK G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. In *Proceedings of SIGGRAPH/Eurographics Symposium on Computer Animation* (2010), SCA '10, Eurographics Association, pp. 217–225. 2, 10

[YT13] YU J., TURK G.: Reconstructing surfaces of particle-based fluids using anisotropic kernels. *ACM Trans. Graph. 32*, 1 (2013), 1–12. 2

[ZB05] ZHU Y., BRIDSON R.: Animating sand as a fluid. *ACM Trans. Graph. 24*, 3 (2005), 965–972. 2

[ZCG14] ZANNI C., CANI M.-P., GLEICHER M.: N-ary implicit blends with topology control. *Computers & Graphics* (October 2014). 3